Institut de Mathématiques d'Orsay
Université Paris-Saclay

Rapport sur les algorithmes d'approximation et de méta-heuristique pour le problème de l'arbre de Steiner

Éric Aubinais, Farid Najar
Master Mathématiques de l'Intelligence Artificielle

Table des matières

0.1 Contexte

Imaginons que nous avons un réseau avec une source et plusieurs destinations. Nous voulons avoir des chemins depuis la source vers les destinations sans être coupé par d'autres chemins. Dit autrement, nous voulons un graphe sans cycles, i.e. un arbre, en partant de la source et en ayant les destinations qu'on veut visiter. De plus, pour aller d'un point à l'autre faut payer un coût. Le but est de trouver l'arbre qui vérifie les conditions énoncées de coût minimal.

0.2 Modélisation

Nous pouvons modéliser le réseau par un graphe connexe $G = (V, E)$ avec V l'ensemble des sommets et E l'ensembles des arrêtes. On introduit aussi une fonction $w : E \to \mathbb{N}$ qui à chaque arrête $e \in E$ attribut un poids $w(e)$. Soit T l'ensemble des "terminaux" qui sont tout simplement les destinations que nous avons énoncé. On remarque que la source peut être considérée comme un terminal. En effet, comme notre solution est un arbre, on peut le représenter comme on veut en prenant un nœud quelconque comme source. Alors notre problème est de trouver un arbre $A = (V', E')$ (graphe connexe sans cycles), tel que, $T \subseteq V'$ et on veut minimiser $\sum_{e \in E'} w(e)$.

0.3 Complexité

0.4 Algorithme d'approximation et taux d'approximation

Comme le problème est NP-Complet, alors si on considère $P \neq NP$, on ne peut construire un algorithme donnant une solution optimale en temps polynomial. Dans ce cas, nous avons plusieurs façon de trouver une solution qui est proche ou, dans certain cas, égal à une solution optimale. Dans ce rapport, nous allons faire et évaluer deux de ces méthodes, "**approximation**" et "**méta-heuristique**". Commençons par l'approximation. Une approximation consiste à essayer, à l'aide de différentes techniques, de trouver la meilleure solution possible en temps polynomial avec une distance maximale déterminée par rapport à la solution optimale qui est appelé le **taux d'approximation** qui est un critère essentiel qui nous permet d'évaluer ce genre d'algorithmes. On peut construire un algorithme d'approximation de différentes manières. Dans cette section, nous allons voir une de ces manières.

0.4.1 Algorithme d'approximation

Nous cherchons un arbre qui passe par tous les sommets terminaux. On remarque que trouver un tel arbre dans un graphe quelconque est équivalent à trouver cet arbre dans un graphe complet avec les terminaux comme sommet. En effet, si on construit ce graphe en mettant la distance minimale entre chaque sommet comme poids de l'arrête qui les relie, et trouver un arbre couvrant de poids minimum qui passe par tous les sommets, on est sûr que nous avons un arbre pour le graphe d'origine en passant par les terminaux. On doit aussi enregistrer les plus courts chemins dans la mémoire afin de ne pas être obligé de les recalculer, et ainsi, ajouter une complexité supplémentaire à notre algorithme. Pour récapituler, nous avons le procédé suivant :

1. On construit G_+ le graphe complet qui a les terminaux comme sommets et sur chaque arrête, on met le poids du plus court chemin. On enregistre, à chaque fois,

le plus court chemin dans la mémoire. Pour calculer les plus courts chemins, on utilise l'algorithme Dijkstra.
Complexité : On pose $n = |V|$. Pour Dijkstra, on a une complexité $O(n^2 \log(n))$. Et pour le stockage des chemins, on a $O(n)$

2. On construit A, l'arbre couvrant de poids minimum de $G+$. Cet algorithme a une complexité de $O(|T| \log(|T|))$.

3. On renvoie l'union des arrêtes qui composent les plus courts chemins. Comme on a enregistré ces chemins, on une complexité $O(1)$.

0.4.2 Taux d'approximation

0.5 Méta-heuristiques

0.5.1 Algorithme recuit

Recuit simple

Recuit multiple

0.5.2 Algorithme génétique

0.6 Résultats et performances